Time-dependent variational forms

Hans Petter Langtangen[1,2]

[1]Center for Biomedical Computing, Simula Research Laboratory
[2]Department of Informatics, University of Oslo

Oct 30, 2015

PRELIMINARY VERSION

Contents

The finite element method is normally used for discretization in space. There are two alternative strategies for performing a discretization in time:

- use finite differences for time derivatives to arrive at a recursive set of spatial problems that can be discretized by the finite element method, or
- discretize in space by finite elements first, and then solve the resulting system of ordinary differential equations (ODEs) by some standard method for ODEs.

We shall exemplify these strategies using a simple diffusion problem

$$\frac{\partial u}{\partial t} = \alpha\nabla^2 u + f(\boldsymbol{x},t), \qquad \boldsymbol{x}\in\Omega,\ t\in(0,T], \tag{1}$$

$$u(\boldsymbol{x},0) = I(\boldsymbol{x}), \qquad \boldsymbol{x}\in\Omega, \tag{2}$$

$$\frac{\partial u}{\partial n} = 0, \qquad \boldsymbol{x}\in\partial\Omega,\ t\in(0,T]\,. \tag{3}$$

Here, $u(\boldsymbol{x},t)$ is the unknown function, α is a constant, and $f(\boldsymbol{x},t)$ and $I(x)$ are given functions. We have assigned the particular boundary condition (3) to minimize the details on handling boundary conditions in the finite element method.

1 Discretization in time by a Forward Euler scheme

1.1 Time discretization

We can apply a finite difference method in time to (1). First we need a mesh in time, here taken as uniform with mesh points $t_n = n\Delta t$, $n = 0,1,\ldots,N_t$. A Forward Euler scheme consists of sampling (1) at t_n and approximating the time derivative by a forward difference $[D_t^+ u]^n \approx (u^{n+1}-u^n)/\Delta t$. This approximation turns (1) into a differential equation that is discrete in time, but still continuous in space. With a finite difference operator notation we can write the time-discrete problem as

$$[D_t^+ u = \alpha\nabla^2 u + f]^n, \tag{4}$$

for $n = 1,2,\ldots,N_t-1$. Writing this equation out in detail and isolating the unknown u^{n+1} on the left-hand side, demonstrates that the time-discrete problem is a recursive set of problems that are continuous in space:

$$u^{n+1} = u^n + \Delta t\left(\alpha\nabla^2 u^n + f(\boldsymbol{x},t_n)\right)\,. \tag{5}$$

Given $u^0 = I$, we can use (5) to compute $u^1, u^2, \ldots, u^{N_t}$.

1.2 Space discretization

We now introduce a finite element approximation to u_{e}^n and u_{e}^{n+1} in (9), where the coefficients depend on the time level:

$$u_{\mathrm{e}}^n \approx u^n = \sum_{j=0}^{N} c_j^n \psi_j(\boldsymbol{x}), \tag{6}$$

$$u_{\mathrm{e}}^{n+1} \approx u^{n+1} = \sum_{j=0}^{N} c_j^{n+1} \psi_j(\boldsymbol{x})\,. \tag{7}$$

Note that, as before, N denotes the number of degrees of freedom in the spatial domain. The number of time points is denoted by N_t. We define a space V spanned by the basis functions $\{\psi_i\}_{i\in\mathcal{I}_s}$.

More precise notation.

For absolute clarity in the various stages of the discretizations, we introduce $u_{\mathrm{e}}(\boldsymbol{x},t)$ as the exact solution of the space-and time-continuous partial differential equation (1) and $u_{\mathrm{e}}^n(\boldsymbol{x})$ as the time-discrete approximation, arising from the finite difference method in time (4). More precisely, u_{e} fulfills

$$\frac{\partial u_{\mathrm{e}}}{\partial t} = \alpha\nabla^2 u_{\mathrm{e}} + f(\boldsymbol{x},t), \tag{8}$$

while u_{e}^{n+1}, with a superscript, is the solution of the time-discrete equations

$$u_{\mathrm{e}}^{n+1} = u_{\mathrm{e}}^n + \Delta t\left(\alpha\nabla^2 u_{\mathrm{e}}^n + f(\boldsymbol{x},t_n)\right)\,. \tag{9}$$

The u_{e}^{n+1} quantity is then discretized in space and approximated by u^{n+1}.

1.3 Variational forms

A Galerkin method or a weighted residual method with weighting functions w_i can now be formulated. We insert (6) and (7) in (9) to obtain the residual

$$R = u^{n+1} - u^n - \Delta t\left(\alpha\nabla^2 u^n + f(\boldsymbol{x},t_n)\right)\,.$$

The weighted residual principle,

$$\int_\Omega Rw\,\mathrm{d}x = 0, \quad \forall w \in W,$$

results in

$$\int_\Omega \left[u^{n+1} - u^n - \Delta t\left(\alpha\nabla^2 u^n + f(\boldsymbol{x}, t_n)\right)\right] w\,\mathrm{d}x = 0, \quad \forall w \in W\,.$$

From now on we use the Galerkin method so $W = V$. Isolating the unknown u^{n+1} on the left-hand side gives

$$\int_\Omega u^{n+1} v\,\mathrm{d}x = \int_\Omega \left[u^n + \Delta t\left(\alpha\nabla^2 u^n + f(\boldsymbol{x}, t_n)\right)\right] v\,\mathrm{d}x, \quad \forall v \in V\,.$$

As usual in spatial finite element problems involving second-order derivatives, we apply integration by parts on the term $\int(\nabla^2 u^n)v\,\mathrm{d}x$:

$$\int_\Omega \alpha(\nabla^2 u^n)v\,\mathrm{d}x = -\int_\Omega \alpha\nabla u^n\cdot\nabla v\,\mathrm{d}x + \int_{\partial\Omega}\alpha\frac{\partial u^n}{\partial n}v\,\mathrm{d}x\,.$$

The last term vanishes because we have the Neumann condition $\partial u^n/\partial n = 0$ for all n. Our discrete problem in space and time then reads

$$\int_\Omega u^{n+1} v\,\mathrm{d}x = \int_\Omega u^n v dx - \Delta t\int_\Omega \alpha\nabla u^n\cdot\nabla v\,\mathrm{d}x + \Delta t\int_\Omega f^n v\,\mathrm{d}x, \quad \forall \boldsymbol{v} \in V\,. \tag{10}$$

This is the variational formulation of our recursive set of spatial problems.

Nonzero Dirichlet boundary conditions.

As in stationary problems, we can introduce a boundary function $B(\boldsymbol{x}, t)$ to take care of nonzero Dirichlet conditions:

$$u_{\mathrm{e}}^n \approx u^n = B(\boldsymbol{x}, t_n) + \sum_{j=0}^{N} c_j^n \psi_j(\boldsymbol{x}), \tag{11}$$

$$u_{\mathrm{e}}^{n+1} \approx u^{n+1} = B(\boldsymbol{x}, t_{n+1}) + \sum_{j=0}^{N} c_j^{n+1} \psi_j(\boldsymbol{x})\,. \tag{12}$$

1.4 Simplified notation for the solution at recent time levels

In a program it is only necessary to store u^{n+1} and u^n at the same time. It is therefore unnatural to use the index n in computer code. Instead a natural variable naming is `u` for u^{n+1}, the new unknown, and `u_1` for u^n, the solution at the previous time level. When we have several preceding (already computed) time levels, it is natural to number them like `u_1`, `u_2`, `u_3`, etc., backwards in time. From this notation in software, we introduce a similar mathematical notation to

help make the mapping from mathematical formulas to implementation as direct as possible. This principle implies that we let u_1 be the discrete unknown at the previous time level (u^n) and u represents the discrete unknown at the new time level (u^{n+1}). Equation (10) with this new naming convention is consequently expressed as

$$\int_\Omega uvdx = \int_\Omega u_1 v dx - \Delta t \int_\Omega \alpha \nabla u_1 \cdot \nabla v \,\mathrm{d}x + \Delta t \int_\Omega f^n v \,\mathrm{d}x\,. \tag{13}$$

This variational form can alternatively be expressed by the inner product notation:

$$(u, v) = (u_1, v) - \Delta t(\alpha \nabla u_1, \nabla v) + \Delta t(f^n, v)\,. \tag{14}$$

1.5 Deriving the linear systems

In the following, we adopt the convention that the unknowns c_j^{n+1} are written as c_j, while the known c_j^n from the previous time level are denoted by $c_{1,j}$. To derive the equations for the new unknown coefficients c_j, we insert

$$u = \sum_{j=0}^{N} c_j \psi_j(\boldsymbol{x}), \quad u_1 = \sum_{j=0}^{N} c_{1,j}\psi_j(\boldsymbol{x})$$

in (13) or (14), let the equation hold for all $v = \psi_i$, $i = 0, \ldots, N$, and order the terms as matrix-vector products:

$$\sum_{j=0}^{N}(\psi_i, \psi_j)c_j = \sum_{j=0}^{N}(\psi_i, \psi_j)c_{1,j} - \Delta t \sum_{j=0}^{N}(\nabla\psi_i, \alpha\nabla\psi_j)c_{1,j} + \Delta t(f^n, \psi_i), \quad i = 0, \ldots, N\,. \tag{15}$$

This is a linear system $\sum_j A_{i,j}c_j = b_i$ with

$$A_{i,j} = (\psi_i, \psi_j)$$

and

$$b_i = \sum_{j=0}^{N}(\psi_i, \psi_j)c_{1,j} - \Delta t \sum_{j=0}^{N}(\nabla\psi_i, \alpha\nabla\psi_j)c_{1,j} + \Delta t(f^n, \psi_i)\,.$$

It is instructive and convenient for implementations to write the linear system on the form

$$Mc = Mc_1 - \Delta t K c_1 + \Delta t f, \tag{16}$$

where

$$
\begin{aligned}
M &= \{M_{i,j}\}, \quad M_{i,j} = (\psi_i, \psi_j), \quad i,j \in \mathcal{I}_s, \\
K &= \{K_{i,j}\}, \quad K_{i,j} = (\nabla\psi_i, \alpha\nabla\psi_j), \quad i,j \in \mathcal{I}_s, \\
f &= \{f_i\}, \quad f_i = (f(\boldsymbol{x}, t_n), \psi_i), \quad i \in \mathcal{I}_s, \\
c &= \{c_i\}, \quad i \in \mathcal{I}_s, \\
c_1 &= \{c_{1,i}\}, \quad i \in \mathcal{I}_s\,.
\end{aligned}
$$

We realize that M is the matrix arising from a term with the zero-th derivative of u, and called the mass matrix, while K is the matrix arising from a Laplace term $\nabla^2 u$. The K matrix is often known as the *stiffness matrix*. (The terms mass and stiffness stem from the early days of finite elements when applications to vibrating structures dominated. The mass matrix arises from the mass times acceleration term in Newton's second law, while the stiffness matrix arises from the elastic forces (the "stiffness") in that law. The mass and stiffness matrix appearing in a diffusion have slightly different mathematical formulas compared to the classic structure problem.)

Remark. The mathematical symbol f has two meanings, either the function $f(\boldsymbol{x}, t)$ in the PDE or the f vector in the linear system to be solved at each time level.

1.6 Computational algorithm

We observe that M and K can be precomputed so that we can avoid computing the matrix entries at every time level. Instead, some matrix-vector multiplications will produce the linear system to be solved. The computational algorithm has the following steps:

1. Compute M and K.
2. Initialize u^0 by interpolation or projection
3. For $n = 1, 2, \ldots, N_t$:
   (a) compute $b = Mc_1 - \Delta t K c_1 + \Delta t f$
   (b) solve $Mc = b$
   (c) set $c_1 = c$

In case of finite element basis functions, interpolation of the initial condition at the nodes means $c_{1,j} = I(\boldsymbol{x}_j)$. Otherwise one has to solve the linear system

$$
\sum_j \psi_j(x_i) c_{1,j} = I(x_i),
$$

where $\boldsymbol{x}_j$ denotes an interpolation point. Projection (or Galerkin's method) implies solving a linear system with M as coefficient matrix:

$$\sum_j M_{i,j} c_{1,j} = (I, \psi_i), \quad i \in \mathcal{I}_s\,.$$

1.7 Example using sinusoidal basis functions

Let us go through a computational example and demonstrate the algorithm from the previous section. We consider a 1D problem

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \qquad x \in (0, L),\ t \in (0, T], \tag{17}$$

$$u(x, 0) = A\cos(\pi \boldsymbol{x}/L) + B\cos(10\pi x/L), \qquad x \in [0, L], \tag{18}$$

$$\frac{\partial u}{\partial x} = 0, \qquad x = 0, L,\ t \in (0, T]\,. \tag{19}$$

We use a Galerkin method with basis functions

$$\psi_i = \cos(i\pi x/L)\,.$$

These basis functions fulfill (19), which is not a requirement (there are no Dirichlet conditions in this problem), but helps to make the approximation good.

Since the initial condition (18) lies in the space V where we seek the approximation, we know that a Galerkin or least squares approximation of the initial condition becomes exact. Therefore,

$$c_{1,1} = A, \quad c_{1,10} = B,$$

while $c_{1,i} = 0$ for $i \neq 1, 10$.

The M and K matrices are easy to compute since the basis functions are orthogonal on $[0, L]$. We get

```
>>> import sympy as sym
>>> x, L = sym.symbols('x L')
>>> i = sym.symbols('i', integer=True)
>>> sym.integrate(sym.cos(i*x*sym.pi/L)**2, (x,0,L))
Piecewise((L, Eq(pi*i/L, 0)), (L/2, True))
```

which means L if $i = 0$ and $L/2$ otherwise. Similarly,

```
>>> sym.integrate(sym.diff(cos(i*x*sym.pi/L),x)**2, (x,0,L))
pi**2*i**2*Piecewise((0, Eq(pi*i/L, 0)), (L/2, True))/L**2
```

so

$$M_{0,0} = L, \quad M_{i,i} = L/2,\ i > 0, \quad K_{0,0} = 0, \quad K_{i,i} = \frac{\pi^2 i^2}{2L},\ i > 0\,.$$

The equation system becomes

$$
\begin{aligned}
Lc_0 &= Lc_{1,0} - \Delta t \cdot 0 \cdot c_{1,0}, \\
\frac{L}{2}c_i &= \frac{L}{2}c_{1,i} - \Delta t \frac{\pi^2 i^2}{2L} c_{1,i}, \quad i > 0\,.
\end{aligned}
$$

The first equation always leads to $c_0 = 0$ since we start with $c_{1,0} = 0$. The others imply

$$
c_i = (1 - \Delta t(\frac{\pi i}{L})^2)c_{1,i}\,.
$$

With the notation c_i^n for c_i at the n-th time level, we can apply the relation above recursively and get

$$
c_i^n = (1 - \Delta t(\frac{\pi i}{L})^2)^n c_i^0\,.
$$

Since only two of the coefficients are nonzero at time $t = 0$, we have the closed-form discrete solution

$$
u_i^n = A(1 - \Delta t(\frac{\pi}{L})^2)^n \cos(\pi x/L) + B(1 - \Delta t(\frac{10\pi}{L})^2)^n \cos(10\pi x/L)\,.
$$

1.8 Comparing P1 elements with the finite difference method

We can compute the M and K matrices using P1 elements in 1D. A uniform mesh on $[0, L]$ is introduced for this purpose. Since the boundary conditions are solely of Neumann type in this sample problem, we have no restrictions on the basis functions ψ_i and can simply choose $\psi_i = \varphi_i$, $i = 0, \ldots, N = N_n - 1$.

From Section 3.2 or 3.4 in [2] we have that the K matrix is the same as we get from the finite difference method: $h[D_x D_x u]_i^n$, while from Section 5.2 in [1] we know that M can be interpreted as the finite difference approximation $h[u + \frac{1}{6}h^2 D_x D_x u]_i^n$. The equation system $Mc = b$ in the algorithm is therefore equivalent to the finite difference scheme

$$
[D_t^+(u + \frac{1}{6}h^2 D_x D_x u) = \alpha D_x D_x u + f]_i^n\,. \tag{20}
$$

(More precisely, $Mc = b$ divided by h gives the equation above.)

Lumping the mass matrix. As explained in Section 5.3 in [1], one can turn the M matrix into a diagonal matrix $\mathrm{diag}(h/2, h, \ldots, h, h/2)$ by applying the Trapezoidal rule for integration. Then there is no need to solve a linear system at each time level, and the finite element scheme becomes identical to a standard finite difference method

$$
[D_t^+ u = \alpha D_x D_x u + f]_i^n\,. \tag{21}
$$

The Trapezoidal integration is not as accurate as exact integration and introduces therefore an error. Normally, one thinks of any error as an overall decrease of the accuracy. Nevertheless, errors may cancel each other, and the error introduced by numerical integration may in certain problems lead to improved overall accuracy in the finite element method. The interplay of the errors in the current problem is analyzed in detail in Section 4. The effect of the error is at least not more severe than what is produced by the finite difference method since both are $\mathcal{O}(h^2)$.

Making M diagonal is usually referred to as *lumping the mass matrix*. There is an alternative method to using an integration rule based on the node points: one can sum the entries in each row, place the sum on the diagonal, and set all other entries in the row equal to zero. For P1 elements the methods of lumping the mass matrix give the same result.

2 Discretization in time by a Backward Euler scheme

2.1 Time discretization

The Backward Euler scheme in time applied to our diffusion problem can be expressed as follows using the finite difference operator notation:

$$[D_t^- u = \alpha\nabla^2 u + f(\boldsymbol{x}, t)]^n \,.$$

Written out, and collecting the unknown u^n on the left-hand side and all the known terms on the right-hand side, the time-discrete differential equation becomes

$$\boldsymbol{u}^n - \Delta t \left(\alpha\nabla^2 \boldsymbol{u}^n + f(\boldsymbol{x}, t_n)\right) = \boldsymbol{u}^{n-1} \,. \tag{22}$$

Equation (22) can compute $\boldsymbol{u}^1, \boldsymbol{u}^2, \ldots, \boldsymbol{u}^{N_t}$, if we have a start $\boldsymbol{u}^0 = I$ from the initial condition. However, (22) is a partial differential equation in space and needs a solution method based on discretization in space. For this purpose we use an expansion as in (6)-(7).

2.2 Variational forms

Inserting (6)-(7) in (22), multiplying by any $v \in V$ (or $\psi_i \in V$), and integrating by parts, as we did in the Forward Euler case, results in the variational form

$$\int_\Omega (u^n v + \Delta t\alpha\nabla u^n \cdot \nabla v)\, \mathrm{d}x = \int_\Omega u^{n-1} v\, \mathrm{d}x + \Delta t \int_\Omega f^n v\, \mathrm{d}x, \quad \forall v \in V \,. \tag{23}$$

Expressed with u as u^n and u_1 as u^{n-1}, the variational form becomes

$$\int_\Omega (uv + \Delta t\alpha\nabla u \cdot \nabla v)\, \mathrm{d}x = \int_\Omega u_1 v\, \mathrm{d}x + \Delta t \int_\Omega f^n v\, \mathrm{d}x, \tag{24}$$

or with the more compact inner product notation,

$$(u, v) + \Delta t(\alpha \nabla u, \nabla v) = (u_1, v) + \Delta t(f^n, v)\,. \tag{25}$$

2.3 Linear systems

Inserting $u = \sum_j c_j \psi_i$ and $u_1 = \sum_j c_{1,j}\psi_i$, and choosing v to be the basis functions $\psi_i \in V$, $i = 0, \ldots, N$, together with doing some algebra, lead to the following linear system to be solved at each time level:

$$(M + \Delta t K)c = Mc_1 + \Delta t f, \tag{26}$$

where M, K, and f are as in the Forward Euler case. This time we really have to solve a linear system at each time level. The computational algorithm goes as follows.

1. Compute M, K, and $A = M + \Delta t K$
2. Initialize u^0 by interpolation or projection
3. For $n = 1, 2, \ldots, N_t$:
   (a) compute $b = Mc_1 + \Delta t f$
   (b) solve $Ac = b$
   (c) set $c_1 = c$

In case of finite element basis functions, interpolation of the initial condition at the nodes means $c_{1,j} = I(\boldsymbol{x}_j)$. Otherwise one has to solve the linear system $\sum_j \psi_j(x_i)c_j = I(x_i)$, where $\boldsymbol{x}_j$ denotes an interpolation point. Projection (or Galerkin's method) implies solving a linear system with M as coefficient matrix: $\sum_j M_{i,j}c_{1,j} = (I, \psi_i)$, $i \in \mathcal{I}_s$.

Finite difference operators corresponding to P1 elements. We know what kind of finite difference operators the M and K matrices correspond to (after dividing by h), so (26) can be interpreted as the following finite difference method:

$$[D_t^-(u + \frac{1}{6}h^2 D_x D_x u) = \alpha D_x D_x u + f]_i^n\,. \tag{27}$$

The mass matrix M can be lumped, as explained in Section 1.8, and then the linear system arising from the finite element method with P1 elements corresponds to a plain Backward Euler finite difference method for the diffusion equation:

$$[D_t^- u = \alpha D_x D_x u + f]_i^n\,. \tag{28}$$

3 Dirichlet boundary conditions

Suppose now that the boundary condition (3) is replaced by a mixed Neumann and Dirichlet condition,

$$u(\boldsymbol{x},t) = u_0(\boldsymbol{x},t), \qquad \boldsymbol{x} \in \partial\Omega_D, \tag{29}$$

$$-\alpha\frac{\partial}{\partial n}u(\boldsymbol{x},t) = g(\boldsymbol{x},t), \qquad \boldsymbol{x} \in \partial\Omega_N\,. \tag{30}$$

Using a Forward Euler discretization in time, the variational form at a time level becomes

$$\int_\Omega u^{n+1}v\,\mathrm{d}x = \int_\Omega (u^n - \Delta t\alpha\nabla u^n\cdot\nabla v)\,\mathrm{d}x + \Delta t\int_\Omega fv\,\mathrm{d}x - \Delta t\int_{\partial\Omega_N} gv\,\mathrm{d}s, \quad \forall v \in V\,. \tag{31}$$

3.1 Boundary function

The Dirichlet condition $u = u_0$ at $\partial\Omega_D$ can be incorporated through a boundary function $B(\boldsymbol{x}) = u_0(\boldsymbol{x})$ and demanding that $v = 0$ at $\partial\Omega_D$. The expansion for u^n is written as

$$u^n(\boldsymbol{x}) = u_0(\boldsymbol{x},t_n) + \sum_{j\in\mathcal{I}_s} c_j^n\psi_j(\boldsymbol{x})\,.$$

Inserting this expansion in the variational formulation and letting it hold for all basis functions ψ_i leads to the linear system

$$\sum_{j\in\mathcal{I}_s}\left(\int_\Omega \psi_i\psi_j\,\mathrm{d}x\right)c_j^{n+1} = \sum_{j\in\mathcal{I}_s}\left(\int_\Omega (\psi_i\psi_j - \Delta t\alpha\nabla\psi_i\cdot\nabla\psi_j)\,\mathrm{d}x\right)c_j^n -$$
$$\int_\Omega (u_0(\boldsymbol{x},t_{n+1}) - u_0(\boldsymbol{x},t_n) + \Delta t\alpha\nabla u_0(\boldsymbol{x},t_n)\cdot\nabla\psi_i)\,\mathrm{d}x$$
$$+\Delta t\int_\Omega f\psi_i\,\mathrm{d}x - \Delta t\int_{\partial\Omega_N} g\psi_i\,\mathrm{d}s, \quad i\in\mathcal{I}_s\,.$$

3.2 Finite element basis functions

When using finite elements, each basis function φ_i is associated with a node x_i. We have a collection of nodes $\{x_i\}_{i\in I_b}$ on the boundary $\partial\Omega_D$. Suppose U_k^n is the known Dirichlet value at x_k at time t_n ($U_k^n = u_0(x_k,t_n)$). The appropriate boundary function is then

$$B(\boldsymbol{x}, t_n) = \sum_{j \in I_b} U_j^n \varphi_j \,.$$

The unknown coefficients c_j are associated with the rest of the nodes, which have numbers $\nu(i)$, $i \in \mathcal{I}_s = \{0, \ldots, N\}$. The basis functions for V are chosen as $\psi_i = \varphi_{\nu(i)}$, $i \in \mathcal{I}_s$, and all of these vanish at the boundary nodes as they should. The expansion for u^{n+1} and u^n become

$$u^n = \sum_{j \in I_b} U_j^n \varphi_j + \sum_{j \in \mathcal{I}_s} c_{1,j} \varphi_{\nu(j)},$$
$$u^{n+1} = \sum_{j \in I_b} U_j^{n+1} \varphi_j + \sum_{j \in \mathcal{I}_s} c_j \varphi_{\nu(j)} \,.$$

The equations for the unknown coefficients $\{c_j\}_{j \in \mathcal{I}_s}$ become

$$\sum_{j \in \mathcal{I}_s} \left(\int_\Omega \varphi_i \varphi_j \,\mathrm{d}x \right) c_j = \sum_{j \in \mathcal{I}_s} \left(\int_\Omega (\varphi_i \varphi_j - \Delta t \alpha \nabla \varphi_i \cdot \nabla \varphi_j) \,\mathrm{d}x \right) c_{1,j} -$$
$$\sum_{j \in I_b} \int_\Omega \left(\varphi_i \varphi_j (U_j^{n+1} - U_j^n) + \Delta t \alpha \nabla \varphi_i \cdot \nabla \varphi_j U_j^n \right) \mathrm{d}x$$
$$+ \Delta t \int_\Omega f \varphi_i \,\mathrm{d}x - \Delta t \int_{\partial\Omega_N} g \varphi_i \,\mathrm{d}s, \quad i \in \mathcal{I}_s \,.$$

3.3 Modification of the linear system

Instead of introducing a boundary function B we can work with basis functions associated with all the nodes and incorporate the Dirichlet conditions by modifying the linear system. Let $\mathcal{I}_s$ be the index set that counts all the nodes: $\{0, 1, \ldots, N = N_n - 1\}$. The expansion for u^n is then $\sum_{j \in \mathcal{I}_s} c_j^n \varphi_j$ and the variational form becomes

$$\sum_{j \in \mathcal{I}_s} \left(\int_\Omega \varphi_i \varphi_j \,\mathrm{d}x \right) c_j = \sum_{j \in \mathcal{I}_s} \left(\int_\Omega (\varphi_i \varphi_j - \Delta t \alpha \nabla \varphi_i \cdot \nabla \varphi_j) \,\mathrm{d}x \right) c_{1,j}$$
$$+ \Delta t \int_\Omega f \varphi_i \,\mathrm{d}x - \Delta t \int_{\partial\Omega_N} g \varphi_i \,\mathrm{d}s \,.$$

We introduce the matrices M and K with entries $M_{i,j} = \int_\Omega \varphi_i \varphi_j \,\mathrm{d}x$ and $K_{i,j} = \int_\Omega \alpha \nabla \varphi_i \cdot \nabla \varphi_j \,\mathrm{d}x$, respectively. In addition, we define the vectors c, c_1, and f with

entries c_i, $c_{1,i}$, and $\int_\Omega f\varphi_i \,\mathrm{d}x - \int_{\partial\Omega_N} g\varphi_i \,\mathrm{d}s$, respectively. The equation system can then be written as

$$Mc = Mc_1 - \Delta t K c_1 + \Delta t f\,. \tag{32}$$

When M, K, and b are assembled without paying attention to Dirichlet boundary conditions, we need to replace equation k by $c_k = U_k$ for k corresponding to all boundary nodes ($k \in I_b$). The modification of M consists in setting $M_{k,j} = 0$, $j \in \mathcal{I}_s$, and the $M_{k,k} = 1$. Alternatively, a modification that preserves the symmetry of M can be applied. At each time level one forms $b = Mc_1 - \Delta t K c_1 + \Delta t f$ and sets $b_k = U_k^{n+1}$, $k \in I_b$, and solves the system $Mc = b$.

In case of a Backward Euler method, the system becomes (26). We can write the system as $Ac = b$, with $A = M + \Delta t K$ and $b = Mc_1 + f$. Both M and K needs to be modified because of Dirichlet boundary conditions, but the diagonal entries in K should be set to zero and those in M to unity. In this way, $A_{k,k} = 1$. The right-hand side must read $b_k = U_k^n$ for $k \in I_b$ (assuming the unknown is sought at time level t_n).

3.4 Example: Oscillating Dirichlet boundary condition

We shall address the one-dimensional initial-boundary value problem

$$u_t = (\alpha u_x)_x + f, \qquad \boldsymbol{x} \in \Omega = [0, L],\ t \in (0, T], \tag{33}$$

$$u(x, 0) = 0, \qquad \boldsymbol{x} \in \Omega, \tag{34}$$

$$u(0, t) = a \sin \omega t, \qquad t \in (0, T], \tag{35}$$

$$u_x(L, t) = 0, \qquad t \in (0, T]\,. \tag{36}$$

A physical interpretation may be that u is the temperature deviation from a constant mean temperature in a body Ω that is subject to an oscillating temperature (e.g., day and night, or seasonal, variations) at $x = 0$.

We use a Backward Euler scheme in time and P1 elements of constant length h in space. Incorporation of the Dirichlet condition at $x = 0$ through modifying the linear system at each time level means that we carry out the computations as explained in Section 2 and get a system (26). The M and K matrices computed without paying attention to Dirichlet boundary conditions become

$$M = \frac{h}{6}\begin{pmatrix} 2 & 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 1 & 4 & 1 & \ddots & & & & & \vdots \\ 0 & 1 & 4 & 1 & \ddots & & & & \vdots \\ \vdots & \ddots & & \ddots & \ddots & 0 & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & 0 & 1 & 4 & 1 & \ddots & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & & \ddots & 1 & 4 & 1 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 & 2 \end{pmatrix} \tag{37}$$

and

$$K = \frac{\alpha}{h}\begin{pmatrix} 1 & -1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ -1 & 2 & -1 & \ddots & & & & & \vdots \\ 0 & -1 & 2 & -1 & \ddots & & & & \vdots \\ \vdots & \ddots & & \ddots & \ddots & 0 & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & 0 & -1 & 2 & -1 & \ddots & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & & \ddots & -1 & 2 & -1 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & -1 & 1 \end{pmatrix} \tag{38}$$

The right-hand side of the variational form contains Mc_1 since there is no source term (f) and no boundary term from the integration by parts ($u_x = 0$ at $x = L$ and we compute as if $u_x = 0$ at $x = 0$ too). We must incorporate the Dirichlet boundary condition $c_0 = a \sin \omega t_n$ by ensuring that this is the first equation in the linear system. To this end, the first row in K and M is set to zero, but the diagonal entry $M_{0,0}$ is set to 1. The right-hand side is $b = Mc_1$, and we set $b_0 = a \sin \omega t_n$. Note that in this approach, $N = N_n - 1$, and c equals the unknown u at each node in the mesh. We can write the complete linear system as

$$c_0 = a\sin\omega t_n, \tag{39}$$

$$\frac{h}{6}(c_{i-1} + 4c_i + c_{i+1}) + \Delta t\frac{\alpha}{h}(-c_{i-1} + 2c_i + c_{i+1}) = \frac{h}{6}(c_{1,i-1} + 4c_{1,i} + c_{1,i+1}), \tag{40}$$

$$i = 1, \ldots, N_n - 2,$$

$$\frac{h}{6}(c_{i-1} + 2c_i) + \Delta t\frac{\alpha}{h}(-c_{i-1} + c_i) = \frac{h}{6}(c_{1,i-1} + 2c_{1,i}), \tag{41}$$

$$i = N_n - 1\,. \tag{42}$$

The Dirichlet boundary condition can alternatively be implemented through a boundary function $B(x,t) = a\sin\omega t\,\varphi_0(x)$:

$$u^n(x) = a\sin\omega t_n\,\varphi_0(x) + \sum_{j\in\mathcal{I}_s} c_j\varphi_{\nu(j)}(x), \quad \nu(j) = j+1\,.$$

Now, $N = N_n - 2$ and the c vector contains values of u at nodes $1, 2, \ldots, N_n - 1$. The right-hand side gets a contribution

$$\int_0^L (a(\sin\omega t_n - \sin\omega t_{n-1})\varphi_0\varphi_i - \Delta t\alpha a\sin\omega t_n\nabla\varphi_0\cdot\nabla\varphi_i)\,\mathrm{d}x\,. \tag{43}$$

4 Analysis of the discrete equations

4.1 Fourier components

The diffusion equation $u_t = \alpha u_{xx}$ allows a (Fourier) wave component

$$u = e^{\beta t + ikx}$$

as solution if $\beta = -\alpha k^2$, which follows from inserting the wave component in the equation ($i = \sqrt{-1}$ is the imaginary unit). This exact wave component can alternatively be written as

$$u = A_{\mathrm{e}}^n e^{ikx}, \quad A_{\mathrm{e}} = e^{-\alpha k^2\Delta t}\,. \tag{44}$$

Many numerical schemes for the diffusion equation have a similar wave component as solution:

$$u_q^n = A^n e^{ikx}, \tag{45}$$

where A is an amplification factor to be calculated by inserting (46) in the scheme. Normally $A \neq A_{\mathrm{e}}$, and the difference in the amplification factor is what introduces (visible) numerical errors. To compute A, we need explicit expressions for the discrete equations for $\{c_j\}_{j\in\mathcal{I}_s}$ in the finite element method. That is, we need to assemble the linear system and look at a general row in the system. This

row can be written as a finite difference scheme, and the analysis of the finite element solution is therefore performed in the same way as for finite difference methods. Expressing the discrete finite element equations as finite difference operators turns out to be very convenient for the calculations.

We introduce $x_q = qh$, or $x_q = q\Delta x$, for the node coordinates, to align the notation with that frequently used in finite difference methods. A convenient start of the calculations is to establish some results for various finite difference operators acting on the wave component

$$u_q^n = A^n e^{ikq\Delta x}. \tag{46}$$

The action of the most common operators of relevance for the model problem at hand are listed below.

$$[D_t^+ A^n e^{ikq\Delta x}]^n = A^n e^{ikq\Delta x}\frac{A-1}{\Delta t}, \tag{47}$$

$$[D_t^- A^n e^{ikq\Delta x}]^n = A^n e^{ikq\Delta x}\frac{1-A^{-1}}{\Delta t}, \tag{48}$$

$$[D_t A^n e^{ikq\Delta x}]^{n+\frac{1}{2}} = A^{n+\frac{1}{2}} e^{ikq\Delta x}\frac{A^{\frac{1}{2}} - A^{-\frac{1}{2}}}{\Delta t} = A^n e^{ikq\Delta x}\frac{A-1}{\Delta t}, \tag{49}$$

$$[D_x D_x A^n e^{ikq\Delta x}]_q = -A^n \frac{4}{\Delta x^2}\sin^2\left(\frac{k\Delta x}{2}\right). \tag{50}$$

4.2 Forward Euler discretization

We insert (46) in the Forward Euler scheme with P1 elements in space and $f = 0$ (note that this type of analysis can only be carried out if $f = 0$),

$$[D_t^+(u + \frac{1}{6}h^2 D_x D_x u) = \alpha D_x D_x u]_q^n. \tag{51}$$

We have

$$[D_t^+ D_x D_x A e^{ikx}]_q^n = [D_t^+ A]^n [D_x D_x e^{ikx}]_q = -A^n e^{ikp\Delta x}\frac{A-1}{\Delta t}\frac{4}{\Delta x^2}\sin^2(\frac{k\Delta x}{2}).$$

The term $[D_t^+ A e^{ikx} + \frac{1}{6}\Delta x^2 D_t^+ D_x D_x A e^{ikx}]_q^n$ then reduces to

$$\frac{A-1}{\Delta t} - \frac{1}{6}\Delta x^2 \frac{A-1}{\Delta t}\frac{4}{\Delta x^2}\sin^2(\frac{k\Delta x}{2}),$$

or

$$\frac{A-1}{\Delta t}\left(1 - \frac{2}{3}\sin^2(k\Delta x/2)\right).$$

Introducing $p = k\Delta x/2$ and $C = \alpha\Delta t/\Delta x^2$, the complete scheme becomes

$$(A-1)\left(1 - \frac{2}{3}\sin^2 p\right) = -4C\sin^2 p,$$

from which we find A to be

$$A = 1 - 4C\frac{\sin^2 p}{1 - \frac{2}{3}\sin^2 p}\,.$$

How does this A change the stability criterion compared to the Forward Euler finite difference scheme and centered differences in space? The stability criterion is $|A| \leq 1$, which here implies $A \leq 1$ and $A \geq -1$. The former is always fulfilled, while the latter leads to

$$4C\frac{\sin^2 p}{1 + \frac{2}{3}\sin^2 p} \leq 2\,.$$

The factor $\sin^2 p/(1 - \frac{2}{3}\sin^2 p)$ can be plotted for $p \in [0, \pi/2]$, and the maximum value goes to 3 as $p \to \pi/2$. The worst case for stability therefore occurs for the shortest possible wave, $p = \pi/2$, and the stability criterion becomes

$$C \leq \frac{1}{6} \quad \Rightarrow \quad \Delta t \leq \frac{\Delta x^2}{6\alpha}, \tag{52}$$

which is a factor 1/3 worse than for the standard Forward Euler finite difference method for the diffusion equation, which demands $C \leq 1/2$. Lumping the mass matrix will, however, recover the finite difference method and therefore imply $C \leq 1/2$ for stability. In other words, introducing an error in the integration improves the stability by a factor of 3.

4.3 Backward Euler discretization

We can use the same approach and insert (46) in the Backward Euler scheme with P1 elements in space and $f = 0$:

$$[D_t^-(u + \frac{1}{6}h^2 D_x D_x u) = \alpha D_x D_x u]_i^n\,. \tag{53}$$

Similar calculations as in the Forward Euler case lead to

$$(1 - A^{-1})\left(1 - \frac{2}{3}\sin^2 p\right) = -4C\sin^2 p,$$

and hence

$$A = \left(1 + 4C\frac{\sin^2 p}{1 - \frac{2}{3}\sin^2 p}\right)^{-1}\,.$$

The quantity in the parentheses is always greater than unity, so $|A| \leq 1$ regardless of the size of C and p. As expected, the Backward Euler scheme is unconditionally stable.

4.4 Comparing amplification factors

It is of interest to compare A and A_{e} as functions of p for some C values. Figure 1 displays the amplification factors for the Backward Euler scheme corresponding to a coarse mesh with $C = 2$ and a mesh at the stability limit of the Forward Euler scheme in the finite difference method, $C = 1/2$. Figures 2 and 3 shows how the accuracy increases with lower C values for both the Forward and Backward Euler schemes, respectively. The striking fact, however, is that the accuracy of the finite element method is significantly less than the finite difference method for the same value of C. Lumping the mass matrix to recover the numerical amplification factor A of the finite difference method is therefore a good idea in this problem.

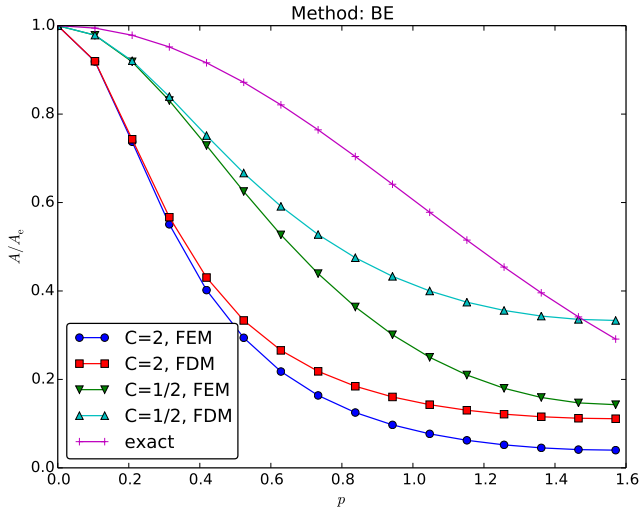

Figure 1: Comparison of coarse-mesh amplification factors for Backward Euler discretization of a 1D diffusion equation.

Remaining tasks:

- Taylor expansion of the error in the amplification factor $A_{\mathrm{e}} - A$
- Taylor expansion of the error $e = (A_{\mathrm{e}}^n - A^n)e^{ikx}$
- L^2 norm of e

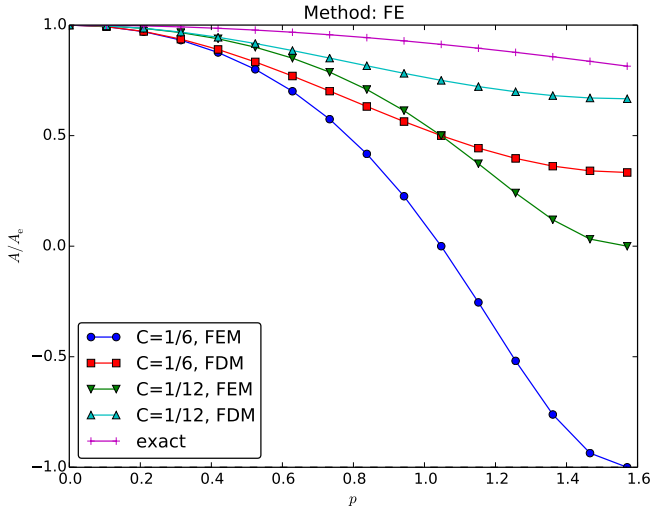

Figure 2: Comparison of fine-mesh amplification factors for Forward Euler discretization of a 1D diffusion equation.

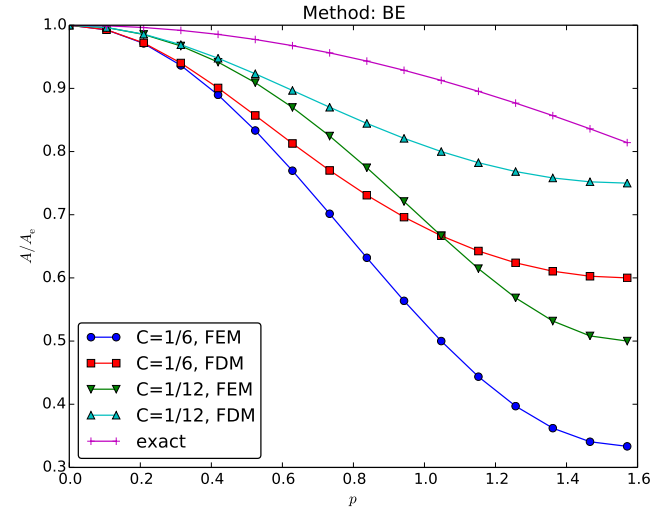

Figure 3: Comparison of fine-mesh amplification factors for Backward Euler discretization of a 1D diffusion equation.

5 Exercises

Exercise 1: Analyze a Crank-Nicolson scheme for the diffusion equation

Perform the analysis in Section 4 for a 1D diffusion equation $u_t = \alpha u_{xx}$ discretized by the Crank-Nicolson scheme in time:

$$\frac{u^{n+1} - u^n}{\Delta t} = \alpha \frac{1}{2}\left(\frac{\partial u^{n+1}}{\partial x^2} + \frac{\partial u^n}{\partial x^2}\right),$$

or written compactly with finite difference operators,

$$[D_t u = \alpha D_x D_x \overline{u}^t]^{n+\frac{1}{2}} .$$

(From a strict mathematical point of view, the u^n and u^{n+1} in these equations should be replaced by u_{e}^n and u_{e}^{n+1} to indicate that the unknown is the exact solution of the PDE discretized in time, but not yet in space, see Section 1.) Make plots similar to those in Section 4. Filename: `fe_diffusion`.

References

[1] H. P. Langtangen. Approximation of functions. `http://tinyurl.com/k3sdbuv/pub/approx`.

[2] H. P. Langtangen. Stationary variational forms. `http://tinyurl.com/k3sdbuv/pub/varform`.

Index